Software und Nachhaltigkeit: Wie Fremdbestimmung durch Software materielle Ressourcen entwertet

Hilty, Lorenz M

Abstract: Die digitale Transformation könnte eine Wirtschaftsweise ermöglichen, die natürliche Ressourcen schont. Software als immateriel-les Produkt kennt keinen Verschleiß. Die universelle Hardware, die zur Ausführung von Software benötigt wird, kommt mit immer weniger Material und Energie pro Leistungseinheit aus. Dennoch wächst der ökologische Fußabdruck der digitalen Transformation. Warum gelingt es uns bisher nicht, Digitalisierung und Nachhaltigkeit in Einklang zu bringen?

Originally published at:
Hilty, Lorenz M (2020). Software und Nachhaltigkeit: Wie Fremdbestimmung durch Software materielle Ressourcen entwertet. FIfF-Kommunikation, (3/2020):31-35.

erschienen in der FIfF-Kommunikation,
herausgegeben von FIfF e.V. - ISSN 0938-3476
www.fiff.de

Lorenz M. Hilty

Software und Nachhaltigkeit

Wie Fremdbestimmung durch Software materielle Ressourcen entwertet

Die digitale Transformation könnte eine Wirtschaftsweise ermöglichen, die natürliche Ressourcen schont. Software als immaterielles Produkt kennt keinen Verschleiß. Die universelle Hardware, die zur Ausführung von Software benötigt wird, kommt mit immer weniger Material und Energie pro Leistungseinheit aus. Dennoch wächst der ökologische Fußabdruck der digitalen Transformation. Warum gelingt es uns bisher nicht, Digitalisierung und Nachhaltigkeit in Einklang zu bringen?

Software als immaterielles und nachhaltiges Produkt

Aus wirtschaftswissenschaftlicher Sicht ist Software ein Informationsgut. Als solches ist Software auf die gleiche Weise immateriell wie ein Roman oder ein Musikstück. Software benötigt zwar, wie alle Informationsgüter, ein materielles Trägermedium, doch dieses ist praktisch vernachlässigbar (Linde 2008).

Im Hinblick auf das Ziel nachhaltiger Produktions- und Konsummuster sind Software-Hardware-Kombinationen auf den ersten Blick ideale Produkte. Veränderte Bedürfnisse lassen sich ohne die Herstellung neuer Hardware erfüllen, denn es muss lediglich die immaterielle Software geändert werden. Software-Produktion ist Wertschöpfung, die nicht auf dem Umsatz von Tonnen von Material, sondern auf der präzisen Formulierung von Gedanken beruht (Hilty 2017). Man könnte also über längere Zeiträume mit der gleichen universellen Hardware leben und dennoch vom Fortschritt in der Software profitieren. Intelligente Steuerung und Regelung durch Software könnte außerdem viele andere material- und energieintensive Prozesse so optimieren, dass sie mit den Prozessen des globalen Ökosystems konsistenter werden. Wenn diese Vision *nicht* Ihrer täglichen Erfahrung im Umgang mit Software-Produkten entspricht, so betrachten Sie sie zunächst einmal als ein optimistisches Szenario.

Hardware als energie- und materialeffizientes Produkt

Im Jahr 1996 hat das deutsche Klimarechenzentrum in Hamburg das weltweit erste Klimamodell gerechnet, das gezeigt hat, dass der beobachtete Klimawandel vom Menschen verursacht ist. Wie viele Smartphones hätte man 25 Jahre später gebraucht, um dieses Modell etwa gleich schnell zu rechnen wie auf dem damaligen Supercomputer? Ein einziges. Im gleichen Zeitraum sind Rechenleistung und Speicherplatz des Hamburger Rechenzentrums übrigens um einen Faktor von rund einer Million gestiegen (DKRZ 2013). Im historischen Trend hat sich die Energieeffizienz der digitalen Hardware alle 1,57 Jahre verdoppelt (Aebischer & Hilty 2015).

Allerdings hat diese erfreuliche Entwicklung auch ihre Schattenseite: In der heutigen digitalen Hardware ist mehr als das halbe Periodensystem enthalten, darunter auch sehr seltene und schwer zu gewinnende Metalle. Wir haben die relative Dematerialisierung dieser Technik also mit einer höheren Materialkomplexität bezahlt. Selbst unter optimalen industriellen Bedingungen werden im Elektronik-Recycling von 50-60 Metallen maximal 17 zurückgewonnen, alle anderen verteilen sich so fein, dass es praktisch nicht mehr möglich ist, sie jemals zurückzuholen. Wir entziehen diese – im Einzelfall winzigen, in der Summe aber relevanten – Metallmengen damit der Nutzung durch zukünftige Generationen (Hilty 2018). Hinzu kommt, dass wir weit davon entfernt sind, optimale industrielle Bedingungen für Recycling überhaupt zu verwirklichen. Selbst in den meisten EU-Ländern klafft zwischen entstehenden und rezyklierten Mengen von Elektronikschrott eine Lücke von über 50 % (Huisman et al. 2017).

Aus Sicht der Nachhaltigkeit wäre es also vernünftig, die digitale Hardware so sparsam und so lange wie möglich zu nutzen und das ursprüngliche Prinzip digitaler Technologie – die Software kommt und geht, die Hardware bleibt – auf breiter Basis zu rehabilitieren. Leider ist der gegenläufige Trend zu beobachten: Software wird dafür eingesetzt, den Konsum von Hardware zu beschleunigen. Materiell hochkomplexe technische Güter werden immer schneller zu Abfall.

Wie Software Hardware entwerten kann

Digitale IKT-Geräte wie z. B. Laptops, Smartphones, Drucker oder Fernseher sind von Software abhängig. Deshalb ist die Versuchung für die Hersteller groß, die Software als Hebel zu nutzen, um die Lebensdauer der Geräte zu begrenzen. Dies sollte uns insbesondere deshalb zu denken geben, weil im Zuge der digitalen Transformation immer mehr Gebrauchsgüter – nicht nur IKT-Geräte – durch eingebettete Mikroprozessoren von Software abhängig werden. Das bedeutet, dass die Beeinflussung der Lebensdauer von Zahnbürsten, Kaffeemaschinen, Kühlschränken, Autos, ja ganzen *Smart Homes* allein durch Software möglich wird. Davon betroffen ist dann nicht nur die eingebettete digitale Elektronik, sondern auch das *Wirtsgut,* also z. B. ein Haushaltgerät, bis hin zu ganzen Infrastrukturen.

Bei der Beeinflussung der Nutzungsdauer von Geräten durch Software ist zu unterscheiden zwischen „programmierter Obsoleszenz" (Brüggemann 2014), die eine Art künstlichen Verschleiß herbeiführt, und „Software-induzierter Obsoleszenz" (Hilty und Aebischer 2015), die Güter durch kontextuelle Faktoren schleichend entwertet.

Bei der *programmierten Obsoleszenz* wird die Lebensdauer eines Geräts oder seine Inkompatibilität mit Ersatzteilen anderer Hersteller durch Instruktionen in der Software explizit beeinflusst. Bekannt geworden sind beispielsweise Fälle, in denen Drucker nach einer bestimmten Zahl von Druckvorgängen versagen. Ebenso gibt es Geräte, die die Ladezyklen ihres Akkus zählen, um diesen dann unabhängig von seinem tatsächlichen chemischen Zustand für gealtert zu erklären. Wenn es dann auch noch praktisch unmöglich ist, den Akku auszutauschen, wird damit ganze Gerät unbrauchbar (Brüggemann 2014). Es handelt sich hier um klassische Fälle geplanter Obsoleszenz – dass sie durch Software realisiert sind, macht sie besonders schwer nachweisbar, weil der Quellcode in der Regel nicht offengelegt wird.

Software-induzierte Obsoleszenz dagegen ist keine geplante Obsoleszenz im herkömmlichen Sinne, da hier keine kausale Beeinflussung der Lebensdauer von Geräten stattfindet. Vielmehr wird die so genannte Evolution der Software genutzt, um Bedingungen herzustellen, die den Betrieb funktionierender Produkte erschweren und diese damit entwerten. Zwar gibt es gute Gründe dafür, dass Software-Produkte nicht statisch sind, sondern sich im Laufe der Zeit weiterentwickeln. Aber die Hersteller unterlassen es in der Regel bewusst, die Kompatibilität mit älteren Versionen eigener (und vor allem fremder) Software und Hardware beizubehalten. Ein häufiger Fall ist die Obsoleszenz von Peripheriegeräten (z. B. Drucker, Scanner, Bildschirme) nach einer Aktualisierung des Betriebssystems, weil der Gerätetreiber zum neuen Betriebssystem nicht mehr kompatibel ist. Ein Gerätetreiber hat die Aufgabe, zwischen dem Betriebssystem eines Computers und dem angeschlossenen Gerät zu vermitteln. Es gibt keine zwingenden technischen Gründe, Standards zu ignorieren und die Schnittstelle zwischen Betriebssystem und Treibern bei einer Aktualisierung des Systems zu ändern. Ist das Problem aber erst einmal geschaffen, kann es nur dadurch gelöst werden, dass jemand eine Version des Treibers entwickelt, die zum neuen Betriebssystem passt. Wenn es niemanden gibt, der dieses kleine Stück Software schreibt, werden mit dem Update schlagartig *alle* in Gebrauch stehenden Geräte obsolet, und das ohne Vorwarnung und ohne Schadenersatz an deren Eigentümer. Auf Nachfrage antworten Hersteller typischerweise, das Gerät werde „nicht mehr unterstützt".

Zur Vermeidung einer solchen *virtuellen Sachbeschädigung* kann man grundsätzlich auf die Aktualisierung des Betriebssystems verzichten. Früher oder später wird man dann aber mit anderen Inkompatibilitäten zu kämpfen haben. Außerdem werden im Laufe der Zeit oft Sicherheitslücken in der älteren Version bekannt. Diese hat der Hersteller durch unprofessionelle Software-Entwicklung zwar selbst verursacht, er schiebt die Verantwortung dafür nun aber jenen Nutzerinnen und Nutzern zu, die die Software nicht „aktuell halten".

Kriterien für nachhaltige Software-Produkte

Im Projekt „*Sustainable Software Design*" des Umweltbundesamtes wurde ein Modell entwickelt, nach dem der Einfluss eines Software-Produkts auf den Energieverbrauch der Hardware, die benötigten Hardware-Kapazitäten und auf die Häufigkeit des Hardware-Wechsels eingeschätzt werden kann. Dabei werden nicht nur das Endgerät, sondern auch die benötigten Übertragungskapazitäten und die in entfernten Rechenzentren verursachten Kapazitätsbedarfe berücksichtigt (Kern et al. 2018). Der Kriterienkatalog umfasst die drei Hauptkriterien Ressourceneffizienz, potenzielle Hardware-Lebensdauer und Nutzungsautonomie (Hilty et al. 2017; Umweltbundesamt 2018).

Die *Ressourceneffizienz* von Software wird daran gemessen, welche Hardware-Kapazitäten sie benötigt (Rechenleistung, Arbeitsspeicher, Permanentspeicher, Bandbreite, Displayauflösung), welchen Energieverbrauch sie in der Hardware verursacht, und in welchem Ausmaß sie fähig ist, Hardware und Energie optimal zu managen. Die nicht messbaren – weil z. B. im Rahmen von Cloud-Diensten beanspruchten – Kapazitäten müssen geschätzt werden.

Die *potenzielle Hardware-Lebensdauer* lässt sich aufgrund der Abwärtskompatibilität der Software mit älterer Hardware und der möglichst weitgehenden Unabhängigkeit von spezifischen Systemumgebungen beurteilen, da dies Freiheitsgrade schafft, die Software auf bereits vorhandener Hardware zu betreiben. Hinzu kommt das Kriterium der Hardware-Suffizienz, das dann erfüllt ist, wenn es dem Hersteller gelingt, die Ansprüche der Software an Hardware-Kapazitäten über die Versionsgeschichte seines Produkts zu verringern (sehr gut), konstant zu halten (gut), oder sie jedenfalls nicht schneller zu steigern, als die technisch bedingte Effizienz voranschreitet (genügend). Wachsen die Ansprüche schneller als die technische Effizienz, ist das Kriterium nicht erfüllt. Wenn ein Software-Produkt von Version zu Version seinen Funktionsumfang erweitert, entlastet dies übrigens nicht von der Forderung nach Hardware-Suffizienz. Es gibt auch einen Fortschritt auf der Ebene der effizienten Software-Gestaltung, der Spielraum schafft für die Realisierung zusätzlicher Funktionen, selbst unter der Bedingung konstanter Hardware-Kapazitäten.

Das dritte Hauptkriterium, *Nutzungsautonomie*, verlangt, dass die Nutzerin oder der Nutzer selbstbestimmt mit dem Software-Produkt umgehen kann und daher unter anderem nicht ge-

drängt wird, mehr Hardware zu konsumieren, als für den intendierten Zweck notwendig ist. Dieses Kriterium wird eingeschätzt aufgrund der Transparenz des Software-Produkts und der verwendeten Datenformate, der Kontinuität des Software-Produkts (lange stabile Phasen ohne Sicherheitsprobleme, möglichst geringe Update-Frequenz), und schließlich auch der Deinstallierbarkeit und der Löschbarkeit der Daten. Letzteres beruht auf der Überlegung, dass es möglich sein sollte, die Nutzung eines Software-Produkts ohne bleibende Spuren einzustellen.

Software-Wirkungen höherer Ordnung

Wir benutzen Software, weil sie bestimmte Handlungen ermöglicht oder unterstützt. Die Wirkungen dieser Handlungen können einen wesentlich stärkeren (positiven oder negativen) Einfluss auf nachhaltige Entwicklung haben als alle energetischen und stofflichen Auswirkungen der Hardware, die für den Betrieb der Software beansprucht wird. Man spricht in diesem Zusammenhang von Effekten höherer Ordnung (Pohl et al. 2018). Nachdem wir uns in den ersten vier Abschnitten dieses Artikels ausschließlich mit Effekten erster Ordnung befasst haben, sollen hier nun die grundlegenden Mechanismen skizziert werden, mit denen die Auswirkungen *der von Software erbrachten Funktionen (enabling impacts)* auf Nachhaltigkeit charakterisiert werden können. Eines der hierfür vorgeschlagenen Modelle differenziert zwischen den folgenden drei Wirkungskategorien (Hilty & Aebischer 2015):

- *Prozessoptimierung:* Hierbei wird ein existierender Prozess durch verbesserten Datenfluss und Informationsverarbeitung näher an sein Optimum gebracht, wobei das Ziel z. B. ein minimaler Energiebedarf sein kann. Beispielsweise kann eine Raumheizung Energie sparen, wenn sie über die An- und Abwesenheit von Personen und deren Präferenzen informiert ist. Im Verkehr kann aktuelle Information über die Auslastung der Kapazitäten von Fahrzeugen und Infrastrukturen Fahrten einsparen.

- *Mediensubstitution:* Ein Trägermedium für Information wird durch ein anderes ersetzt; abhängig von der materiellen Beschaffenheit der Trägermedien und der Form der Nutzung kann dies unter Nachhaltigkeitsaspekten von Vorteil oder von Nachteil sein. Beispielsweise spart der Ersatz von bedrucktem Papier durch elektronische Dokumente nur dann Ressourcen, wenn die Endgeräte für mehrere Zwecke oder sehr lange genutzt werden. Videokonferenzen sind im Vergleich zu Flugreisen eindeutig nachhaltiger (Coroama et al. 2015; Warland et al. 2016).

- *Externalisierung von Kontrolle:* Wird ein Prozess durch Software gesteuert, so kann diese Kontrolle ortsunabhängig ausgeübt werden. Dadurch ist beispielsweise die Wartung von Maschinen oder die Überwachung und Steuerung von Prozessen auf Distanz möglich. Dieser Effekt hat ein erhebliches Potenzial zur Verringerung des Ressourcenbedarfs. Er bringt aber auch Risiken der Fremdbestimmung mit sich, auf die ich unten näher eingehen werde.

In allen drei Fällen ist zu bedenken, dass die Einsparung von Ressourcen durch Rebound-Effekte kompensiert werden kann: Wird eine Leistung billiger oder schneller zur Verfügung gestellt, regt dies die Nachfrage an, so dass die technisch ermöglichte Einsparung von Ressourcen teilweise oder vollständig kompensiert, im Extremfall auch überkompensiert wird (Hilty et al. 2006).

Unabhängig von Rebound-Effekten birgt aber die dritte Wirkungskategorie, Externalisierung von Kontrolle, auch spezifische Risiken, indem sie zu einer zunehmenden Fremdbestimmung von Gebrauchsgütern führt. Einer der Haupttrends der Digitalisierung besteht bekanntlich darin, dass mehr und mehr Dinge des Alltags (nicht nur IKT-Geräte) durch eingebettete Prozessoren *smart*, also durch Software gesteuert und häufig auch internetfähig werden. Dies betrifft z. B. Haushaltgeräte, Spielsachen, Einrichtungsgegenstände, Kleidung und Fahrzeuge. Dieser Trend wurde in seinen Anfängen vor 20 Jahren als *Pervasive Computing*, *Ubiquitous Computing* oder auch *Ambient Intelligence* bezeichnet, heute sind die Bezeichnungen *Internet der Dinge,*

Plakat in der Ausstellung Hello, Robot, Museum für angewandte Kunst Wien 2017 – Foto privat

Smart Everything oder *Cyber-Physical Systems* in Gebrauch. Die semantischen Abstufungen dieser Buzzwords sind irrelevant für den hier betrachteten zentralen Aspekt: Software gewinnt Kontrolle über materielle Vorgänge, was die Frage aufwirft, wer diese Kontrolle ausübt. Denn Software trifft Entscheidungen nach Regeln, die bei der Entwicklung der Software festgelegt wurden. Als Besitzer eines *smarten* Dings sollte mich interessieren, wer diese Regeln festlegt, ändern kann und in wessen Interesse dies geschieht.

Informationsasymmetrie, Qualitätsprobleme und Fremdbestimmung

Von Informationsasymmetrie spricht man, wenn die Information über ein Gut zwischen Anbieter- und Nachfragerseite ungleich verteilt ist. Eine solche Asymmetrie ist bei allen Informationsgütern gegeben, da ein vollständiger Einblick in ein auf dem Markt angebotenes Informationsgut dessen Erwerb überflüssig machen würde (Informationsparadox).

Bei Software ist die Informationsasymmetrie normalerweise besonders hoch, da die Anbietenden sehr viel mehr Information über das Produkt besitzten als die Nachfragenden. Selbst bei Offenlegung der Quellprogramme sind hoher Aufwand und spezielles Fachwissen erforderlich, um daraus Produkteigenschaften abzuleiten.

Software-Produkte fallen deshalb in die Kategorie der *Erfahrungsgüter* und der *Vertrauensgüter:* Die Nachfragenden haben praktisch keine Möglichkeit, sich im Voraus über die Qualität des Gutes zu informieren, sie können sich lediglich (wie z. B. auch bei der Dienstleistung eines Arztes) auf eigene oder fremde Erfahrung verlassen und müssen den Anbietenden in einem gewissen Ausmaß vertrauen. Für Anbieter von Gütern mit ausgeprägten Erfahrungs- und Vertrauenseigenschaften bieten sich generell sehr weitgehende Möglichkeiten zu strategischem Verhalten (Linde 2008).

Informationsasymmetrie hat primär den Effekt, dass auf dem jeweiligen Markt gute Qualität durch schlechte Qualität verdrängt wird. Es überrascht daher nicht, dass auf dem Software-Markt fehlerhafte und unsichere Produkte die Regel sind. Die meisten Nutzerinnen und Nutzer haben sich an diesen Zustand gewöhnt und ahnen nicht, welches Ausmaß an Unprofessionalität und Zynismus den Alltag der kommerziellen Software-Entwicklung beherrscht.

Dies zeigt sich in Fällen, in denen von Anbietern die Offenlegung des Quellcodes verlangt wird und dieser analysiert werden kann. Ein Beispiel ist das *E-Voting*-System, das die schweizerische Post zusammen mit der Software-Firma *Scytl* entwickelt hat. Experten, die das Produkt analysierten, stellten nicht nur eine Reihe schwerwiegender handwerklicher Mängel fest, sogar der schlimmste denkbare Mangel blieb nicht aus: Das E-Voting-System würde in seiner jetzigen Form die Manipulation der Ergebnisse erlauben. „Personen, die Zugang zum Server mit den Ergebnissen haben, um diese zu verifizieren, hätten bei diesem Prozess die Ergebnisse unbemerkt manipulieren können" (Kolly 2019). Die Tatsache, dass der Hersteller auf dieses Problem bereits zwei Jahre zuvor aufmerksam gemacht wurde und es nicht behoben hat, spricht für sich. Gibt es einen direkteren Weg, eine funktionierende Demokratie zu beschädigen, als sie mit manipulierbarer Software zu versorgen?

Betrachten wir das Problem der Informationsasymmetrie bei Software-Produkten nun im Zusammenhang mit dem erwähnten Trend, dass die Kontrolle über einen wachsenden Teil unserer Alltagsgüter und Infrastrukturen externalisiert wird. Über die Qualitätsprobleme hinaus entsteht bei Software folglich das Problem, dass der Anbieter der Software Kontrolle über die davon abhängigen Güter ausüben kann auf eine Weise, die für deren Eigentümer nicht transparent ist.

Davon bildet die programmierte Obsoleszenz einen wichtigen, aber nicht den einzigen Spezialfall. In anderen Fällen wird Software genutzt, nicht gegebene Produkteigenschaften vorzutäuschen (wie dies bei Diesel-PKW und bei Kühlschränken der Fall war), Konkurrenzprodukte zu benachteiligen (was bei Druckerpatronen und anderen Ersatzteilen nachgewiesen wurde) oder Nachfrage nach eigenen Produkten zu schaffen. Generell schafft die Externalisierung von Kontrolle über materielles Eigentum günstige Voraussetzungen zur Schaffung von Märkten, in denen die Anbieter die Nachfrage selbst steuern können. Dass unter solchen Bedingungen nachhaltige Muster von Produktion und Konsum entstehen, erscheint wenig wahrscheinlich.

Von der informationellen zur materiellen Selbstbestimmung

Das Recht auf informationelle Selbstbestimmung wurde vom deutschen Verfassungsgericht aus dem allgemeinen Persönlichkeitsrecht heraus entwickelt, um den Einzelnen eine rechtliche Handhabe zu geben, über die Verwendung der persönlichen Daten zu bestimmen. Wie wir gesehen haben, schwächt die Digitalisierung aber nicht nur die Kontrolle über die eigenen Daten, sondern zunehmend auch die Kontrolle über materielles Eigentum.

Es stellt sich deshalb die Frage, ob nicht aus dem Eigentumsrecht ein „Recht auf materielle Selbstbestimmung" abzuleiten wäre,

Lorenz Hilty

Lorenz Hilty ist seit 2010 Professor für Informatik und Nachhaltigkeit am Institut für Informatik der Universität Zürich. Zuvor leitete er die Abteilung *Technologie und Gesellschaft* der Eidgenössischen Materialforschungsanstalt in St. Gallen, Schweiz. 2013 initiierte er die internationale Konferenzreihe *ICT for Sustainability (ICT4S)*.

das den Eigentümern eines (Software-gesteuerten) materiellen Gutes bessere Möglichkeiten gibt, sich gegen die Untergrabung ihrer Verfügungsgewalt über dieses Gut zu schützen.

Ein denkbarer Ansatz hierfür wäre das in den USA für Autos geltende und für Elektronikprodukte derzeit geforderte *„Right to Repair"* (Reichwein & Sydow 2018). Ein Recht auf Reparatur würde sich bei Produkten mit Software-Anteil auch auf die Software beziehen müssen, die folglich offengelegt werden müsste. Allein dieser Aspekt könnte sich heilsam auf die Kultur der Software-Entwicklung auswirken.

Fazit

Durch Fortschritte im Bereich der Software in Verbindung mit langlebiger, universeller Hardware könnte eine ressourcenschonende Form des Wirtschaftens realisiert werden. Die digitale Transformation führt bisher nicht in diese nachhaltige Richtung. Zentrale Gründe hierfür sind:

- Rebound-Effekte innerhalb und außerhalb des IKT-Sektors,
- Informationsasymmetrien im Software-Markt, die zu Qualitätsproblemen führen und außerdem geplante und induzierte Obsoleszenz begünstigen,
- das zunehmende Risiko eines Verlusts der Verfügungsgewalt über materielles Eigentum, das von Software abhängig ist.

Rebound-Effekte sind nur zu verhindern, wenn sich die Knappheit natürlicher Ressourcen (einschließlich der Entsorgungskapazitäten von Senken) in ihren Preisen widerspiegelt.

Die Undurchschaubarkeit von Software-Produkten kann dadurch gelindert werden, dass überprüfbare Kriterien für nachhaltige Software und entsprechende Kennzeichnungen eingeführt werden.

Gegen die zunehmende Fremdbestimmung von Gebrauchsgütern sollten wir alle durch ein „Recht auf materielle Selbstbestimmung", analog dem Recht auf informationelle Selbstbestimmung, besser geschützt werden.

Referenzen

Aebischer, Bernard, Hilty, L. M.: The Energy Demand of ICT: A Historical Perspective and Current Methodological Challenges. In: ICT Innovations for Sustainability. Springer, Cham 2015 71–103. https://doi.org/10.5167/uzh-110003

Brüggemann, Salim: Geplante Obsoleszenz IT-basierter Geräte. Universität Zürich 2014 https://www.merlin.uzh.ch/publication/show/10511

Coroama, Vlad C., Moberg, Å., Hilty, L. M.: Dematerialization through electronic media? In: ICT Innovations for Sustainability. Springer, Cham 2015, 405–421 https://doi.org/10.5167/uzh-109996

DKRZ, 25 Jahre Deutsches Klimarechenzentrum. Hamburg 2013 https://www.dkrz.de/kommunikation/pressemitteilungen/25-jahre-deutsches-klimarechenzentrum

Engelhardt, Sebastian von: Die ökonomischen Eigenschaften von Software. Jena 2006 http://www2.wiwi.uni-jena.de/Papers/wp-sw1406.pdf

Hilty, Lorenz M., Aebischer, B.: ICT for Sustainability: an Emerging Research Field. In: ICT Innovations for Sustainability. Springer, Cham 2015, 3–36 https://doi.org/10.5167/uzh-110001

Hilty, Lorenz M., Grundlagenforschung in der Informatik? Perspektiven der Informatik und ihre Erkenntnisziele. Vereinigung der Schweizerischen Hochschuldozierenden, 2017 3-10 https://doi.org/10.5167/uzh-141516

Hilty, Lorenz M., Internal Error: Systemdenken fehlt. Green IT im Kontext der Digitalisierung. Politische Ökologie, 155/2018, 33-38 https://files.ifi.uzh.ch/hilty/p/2018_Hilty_Internal-Error-Systemdenken-fehlt.pdf

Hilty, Lorenz M.; Köhler, Andreas; von Schéele, Fabian; Zah, Rainer; Ruddy, Thomas: Rebound Effects of Progress in Information Technology. International Journal of Technology Assessment and Ethics of Science, 1 (4) 2006, 19-38 https://doi.org/10.1007/s10202-005-0011-2

Hilty, Lorenz M.; Naumann, Stefan; Gröger, Jens et al. Kriterienkatalog für nachhaltige Software. Birkenfeld 2017 http://green-software-engineering.de/kriterienkatalog

Huisman, Jaco; Leroy, Pascal; Tertre, Francois; Ljunggren Söderman, Maria; Chancerel, Perrine; Cassard, Daniel; Løvik, Amund; Wäger, Patrick; Kushnir, Duncan; Rotter, Vera; Mählitz, Paul; Herreras, Lucía; Emmerich, Johanna; Hallberg, Anders; Habib, Hina; Wagner, Michelle; Downes, Sarah. Prospecting Secondary Raw Materials in the Urban Mine and mining wastes (ProSUM). Final Report. Brussels 2017 https://doi.org/10.13140/RG.2.2.10451.89125

Kern, Eva; Hilty, Lorenz M.; Guldner, Achim; Maksimov, Yuliyan; Filler, Andreas; Gröger, Jens; Naumann, Stefan: Sustainable software products –Towards assessment criteria for resource and energy efficiency. Future Generation Computer Systems 86, 2018, 199-210 https://doi.org/10.1016/j.future.2018.02.044

Linde, Frank: Ökonomie der Information. Göttingen 2008 https://doi.org/10.17875/gup2008-212

Pohl, Johanna; Hilty, Lorenz M., Finkbeiner, Martin: How LCA contributes to the environmental assessment of higher order effects of ICT application: A review of different approaches. Journal of Cleaner Production, Volume 219, 10 May 2019, 698-712 https://doi.org/10.1016/j.jclepro.2019.02.018

Reichwein, Antonia; Sydow, Johanna: Wege aus der Reparaturkrise? Germanwatch. Berlin 2018 https://germanwatch.org/de/15871

Umweltbundesamt: Entwicklung und Anwendung von Bewertungsgrundlagen für ressourceneffiziente Software unter Berücksichtigung bestehender Methodik. Berlin, 2018 https://www.umweltbundesamt.de/sites/default/files/medien/1410/publikationen/2018-12-12_texte_105-2018_ressourceneffiziente-software_0.pdf

Warland, Linde; Hilty, L.; Küng, J.; Reinhard, J.: Factsheet: Dienstreisen. Universität Zürich 2016 https://www.sustainability.uzh.ch/de/Factsheets-und-Empfehlungen/Factsheets

Diesen Beitrag haben wir übernommen aus:
Die Ökologie der digitalen Gesellschaft. Jahrbuch Ökologie 2019/2020, Hirzel, Stuttgart 2019, und etwas gekürzt.

Wir danken dem Verlag für die Genehmigung zum Nachdruck.